

William Stallings

Data and Computer

Communications

7th Edition

Chapter 12

Routing

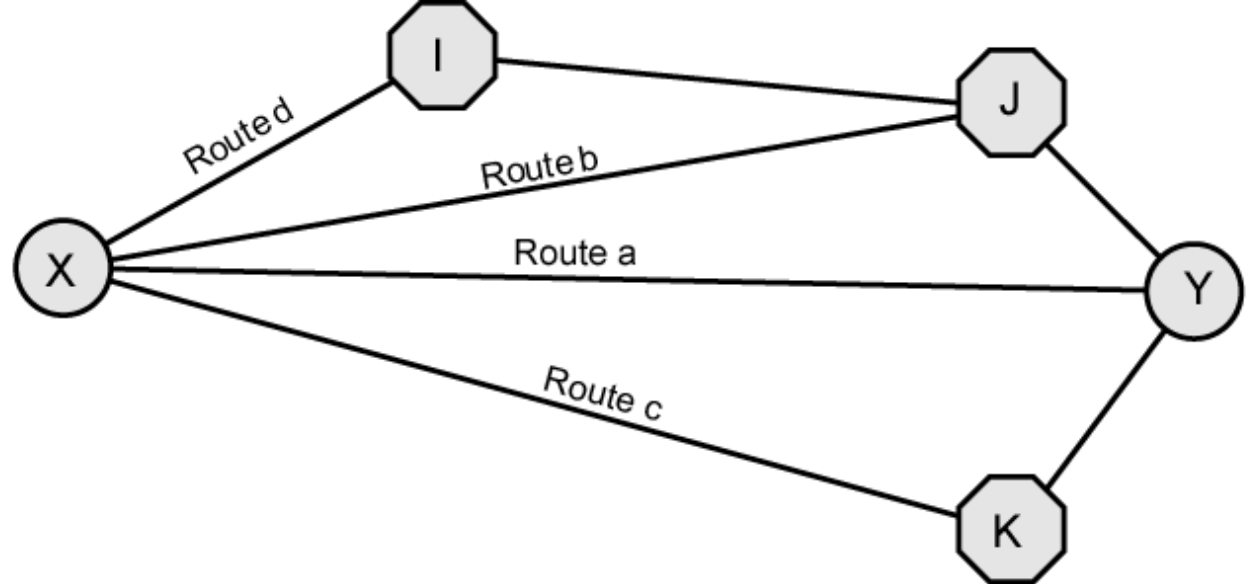
Routing in Circuit Switched Network

- Many connections will need paths through more than one switch
- Need to find a route
 - Efficiency
 - Resilience
- Public telephone switches are a tree structure
 - Static routing uses the same approach all the time
- Dynamic routing allows for changes in routing depending on traffic
 - Uses a peer structure for nodes

Alternate Routing

- Possible routes between end offices predefined
- Originating switch selects appropriate route
- Routes listed in preference order
- Different sets of routes may be used at different times

Alternate Routing Diagram



Route a: X® Y
Route b: X® J® Y
Route c: X® K® Y
Route d: X® I® J® Y

○ = end office

⬡ = intermediate switching node

(a) Topology

Time Period	First route	Second route	Third route	Fourth and final route
Morning	a	b	c	d
Afternoon	a	d	b	c
Evening	a	d	c	b
Weekend	a	c	b	d

(b) Routing table

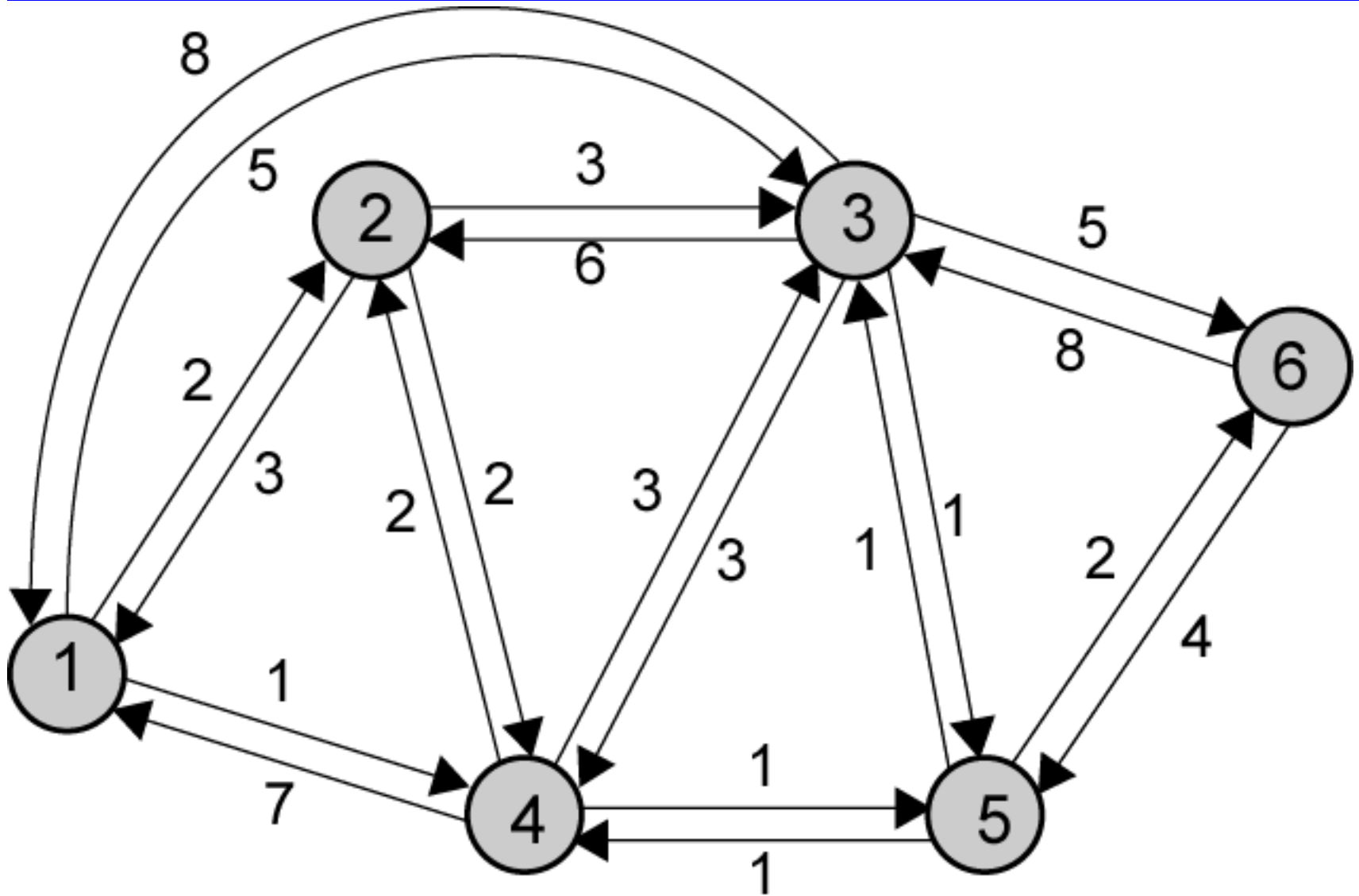
Routing in Packet Switched Network

- Complex, crucial aspect of packet switched networks
- Characteristics required
 - Correctness
 - Simplicity
 - Robustness
 - Stability
 - Fairness
 - Optimality
 - Efficiency

Performance Criteria

- Used for selection of route
- Minimum hop
- Least cost
 - See Stallings appendix 10A for routing algorithms

Example Packet Switched Network



Decision Time and Place

- Time
 - Packet or virtual circuit basis
- Place
 - Distributed
 - Made by each node
 - Centralized
 - Source

Network Information Source and Update Timing

- Routing decisions usually based on knowledge of network (not always)
- Distributed routing
 - Nodes use local knowledge
 - May collect info from adjacent nodes
 - May collect info from all nodes on a potential route
- Central routing
 - Collect info from all nodes
- Update timing
 - When is network info held by nodes updated
 - Fixed - never updated
 - Adaptive - regular updates

Routing Strategies

- Fixed
- Flooding
- Random
- Adaptive

Fixed Routing

- Single permanent route for each source to destination pair
- Determine routes using a least cost algorithm (appendix 10A)
- Route fixed, at least until a change in network topology

Fixed Routing Tables

CENTRAL ROUTING DIRECTORY

		From Node					
		1	2	3	4	5	6
To Node	1	—	1	5	2	4	5
	2	2	—	5	2	4	5
	3	4	3	—	5	3	5
	4	4	4	5	—	4	5
	5	4	4	5	5	—	5
	6	4	4	5	5	6	—

Node 1 Directory

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Node 2 Directory

Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Node 3 Directory

Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Node 4 Directory

Destination	Next Node
1	2
2	2
3	5
5	5
6	5

Node 5 Directory

Destination	Next Node
1	4
2	4
3	3
4	4
6	6

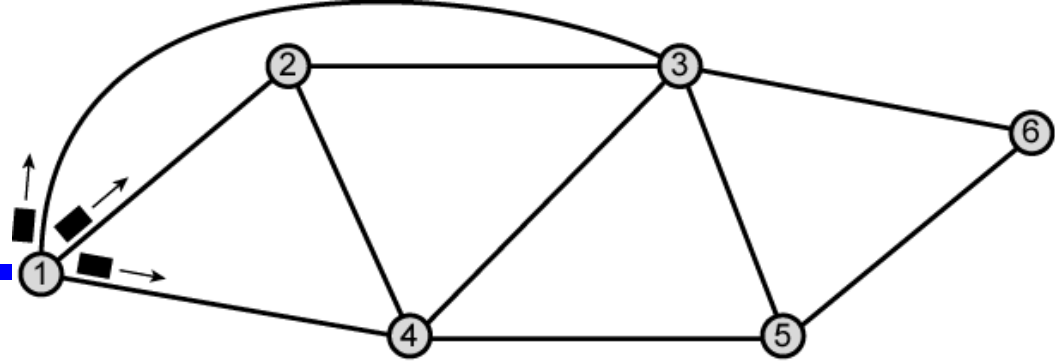
Node 6 Directory

Destination	Next Node
1	5
2	5
3	5
4	5
5	5

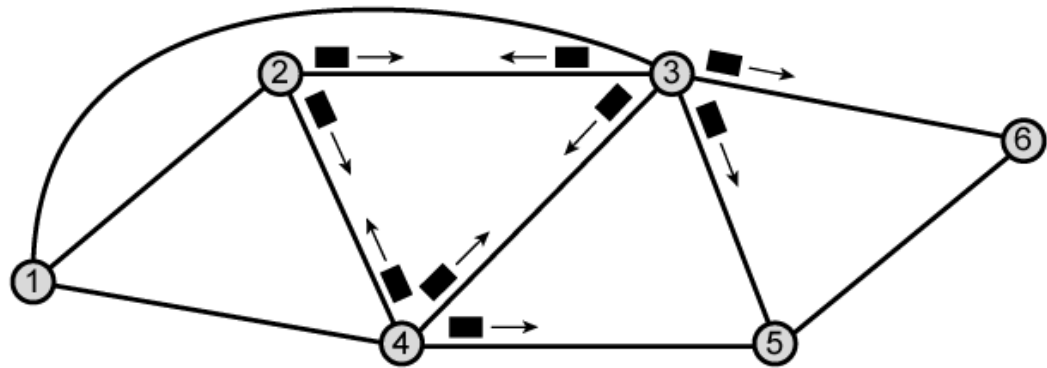
Flooding

- No network info required
- Packet sent by node to every neighbor
- Incoming packets retransmitted on every link except incoming link
- Eventually a number of copies will arrive at destination
- Each packet is uniquely numbered so duplicates can be discarded
- Nodes can remember packets already forwarded to keep network load in bounds
- Can include a hop count in packets

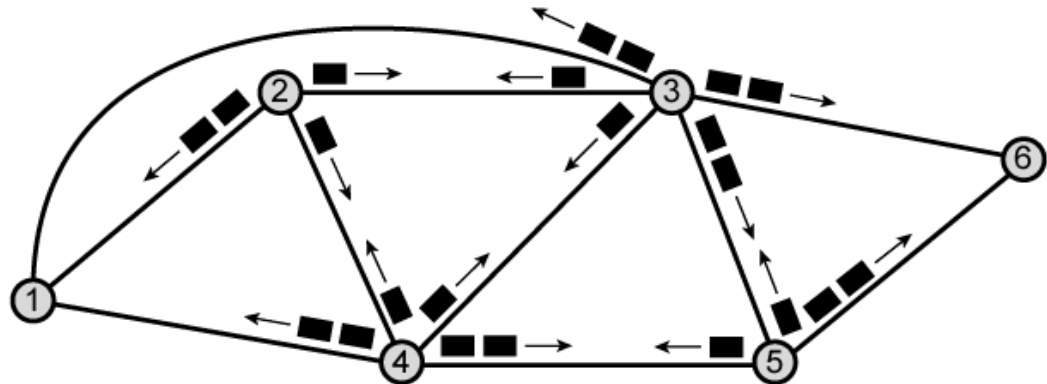
Flooding Example



(a) First hop



(b) Second hop



(c) Third hop

Properties of Flooding

- All possible routes are tried
 - Very robust
- At least one packet will have taken minimum hop count route
 - Can be used to set up virtual circuit
- All nodes are visited
 - Useful to distribute information (e.g. routing)

Random Routing

- Node selects one outgoing path for retransmission of incoming packet
- Selection can be random or round robin
- Can select outgoing path based on probability calculation
- No network info needed
- Route is typically not least cost nor minimum hop

Adaptive Routing

- Used by almost all packet switching networks
- Routing decisions change as conditions on the network change
 - Failure
 - Congestion
- Requires info about network
- Decisions more complex
- Tradeoff between quality of network info and overhead
- Reacting too quickly can cause oscillation
- Too slowly to be relevant

Adaptive Routing - Advantages

- Improved performance
- Aid congestion control (See chapter 13)
- Complex system
 - May not realize theoretical benefits

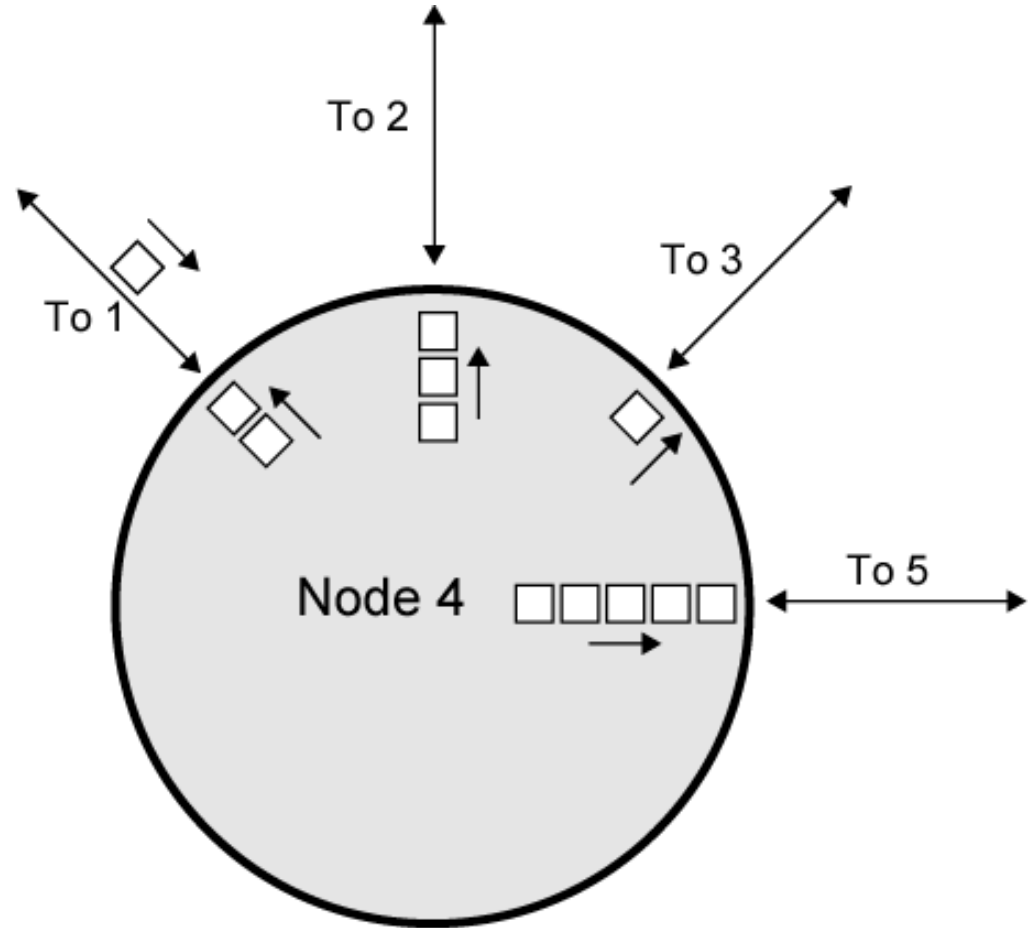
Classification

- Based on information sources
 - Local (isolated)
 - Route to outgoing link with shortest queue
 - Can include bias for each destination
 - Rarely used - do not make use of easily available info
 - Adjacent nodes
 - All nodes

Isolated Adaptive Routing

Node 4's Bias
Table for
Destination 6

Next Node	Bias
1	9
2	6
3	3
5	0



ARPANET Routing Strategies(1)

- First Generation
 - 1969
 - Distributed adaptive
 - Estimated delay as performance criterion
 - Bellman-Ford algorithm (appendix 10a)
 - Node exchanges delay vector with neighbors
 - Update routing table based on incoming info
 - Doesn't consider line speed, just queue length
 - Queue length not a good measurement of delay
 - Responds slowly to congestion

ARPANET Routing Strategies(2)

- Second Generation
 - 1979
 - Uses delay as performance criterion
 - Delay measured directly
 - Uses Dijkstra's algorithm (appendix 10a)
 - Good under light and medium loads
 - Under heavy loads, little correlation between reported delays and those experienced

ARPANET Routing Strategies(3)

- Third Generation
 - 1987
 - Link cost calculations changed
 - Measure average delay over last 10 seconds
 - Normalize based on current value and previous results

Least Cost Algorithms

- Basis for routing decisions
 - Can minimize hop with each link cost 1
 - Can have link value inversely proportional to capacity
- Given network of nodes connected by bi-directional links
- Each link has a cost in each direction
- Define cost of path between two nodes as sum of costs of links traversed
- For each pair of nodes, find a path with the least cost
- Link costs in different directions may be different
 - E.g. length of packet queue

Dijkstra's Algorithm Definitions

- Find shortest paths from given source node to all other nodes, by developing paths in order of increasing path length
- N = set of nodes in the network
- s = source node
- T = set of nodes so far incorporated by the algorithm
- $w(i, j)$ = link cost from node i to node j
 - $w(i, i) = 0$
 - $w(i, j) = \infty$ if the two nodes are not directly connected
 - $w(i, j) \geq 0$ if the two nodes are directly connected
- $L(n)$ = cost of least-cost path from node s to node n currently known
 - At termination, $L(n)$ is cost of least-cost path from s to n

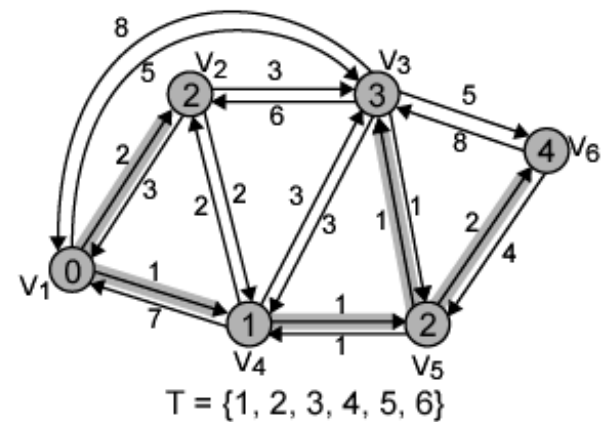
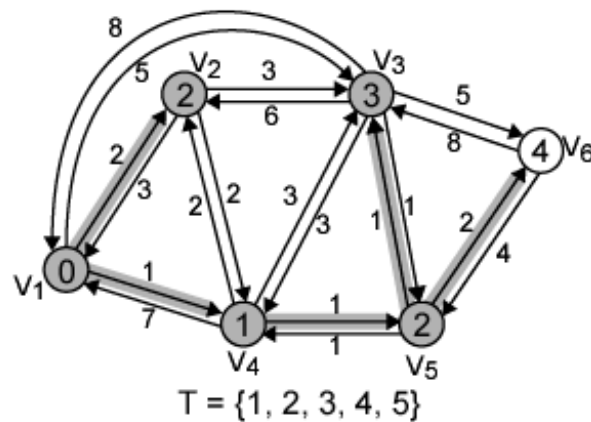
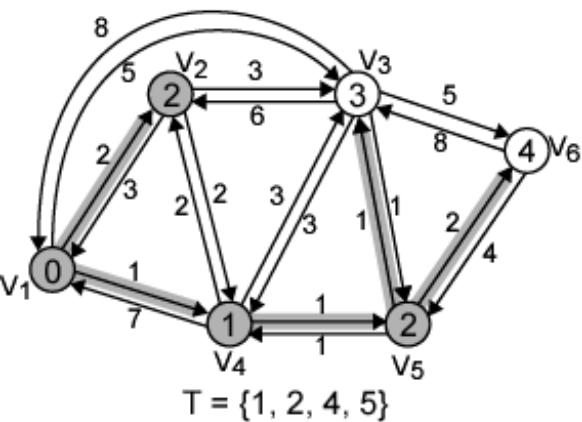
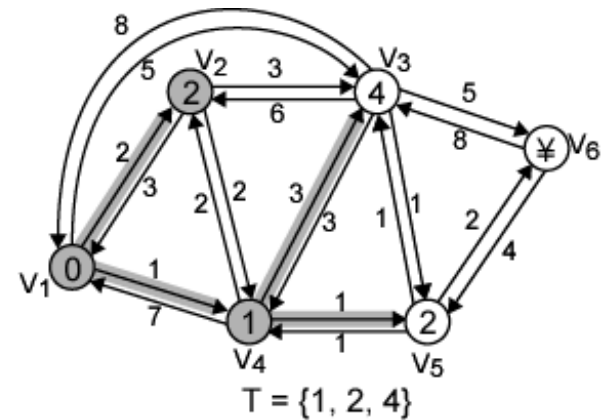
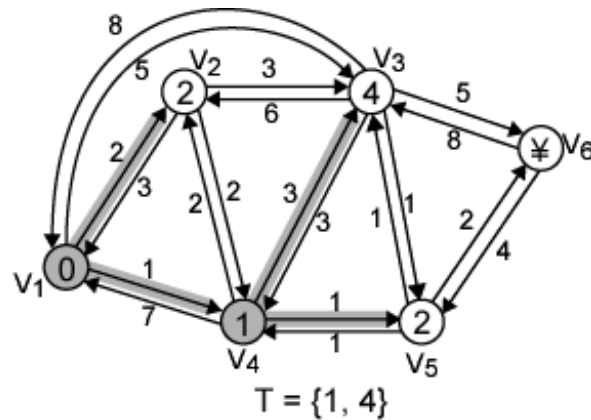
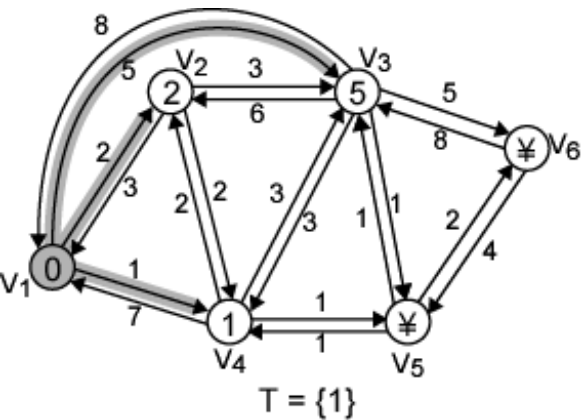
Dijkstra's Algorithm Method

- Step 1 [Initialization]
 - $T = \{s\}$ Set of nodes so far incorporated consists of only source node
 - $L(n) = w(s, n)$ for $n \neq s$
 - Initial path costs to neighboring nodes are simply link costs
- Step 2 [Get Next Node]
 - Find neighboring node not in T with least-cost path from s
 - Incorporate node into T
 - Also incorporate the edge that is incident on that node and a node in T that contributes to the path
- Step 3 [Update Least-Cost Paths]
 - $L(n) = \min[L(n), L(x) + w(x, n)]$ for all $n \notin T$
 - If latter term is minimum, path from s to n is path from s to x concatenated with edge from x to n
- Algorithm terminates when all nodes have been added to T

Dijkstra's Algorithm Notes

- At termination, value $L(x)$ associated with each node x is cost (length) of least-cost path from s to x .
- In addition, T defines least-cost path from s to each other node
- One iteration of steps 2 and 3 adds one new node to T
 - Defines least cost path from s to that node

Example of Dijkstra's Algorithm



Results of Example Dijkstra's Algorithm

Iteration	T	L(2)	Path	L(3)	Path	L(4)	Path	L(5)	Path	L(6)	Path
1	{1}	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	{1,4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
3	{1, 2, 4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
4	{1, 2, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
5	{1, 2, 3, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
6	{1, 2, 3, 4, 5, 6}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

Bellman-Ford Algorithm

Definitions

- Find shortest paths from given node subject to constraint that paths contain at most one link
- Find the shortest paths with a constraint of paths of at most two links
- And so on
- s = source node
- $w(i, j)$ = link cost from node i to node j
 - $w(i, i) = 0$
 - $w(i, j) = \infty$ if the two nodes are not directly connected
 - $w(i, j) \geq 0$ if the two nodes are directly connected
- h = maximum number of links in path at current stage of the algorithm
- $L_h(n)$ = cost of least-cost path from s to n under constraint of no more than h links

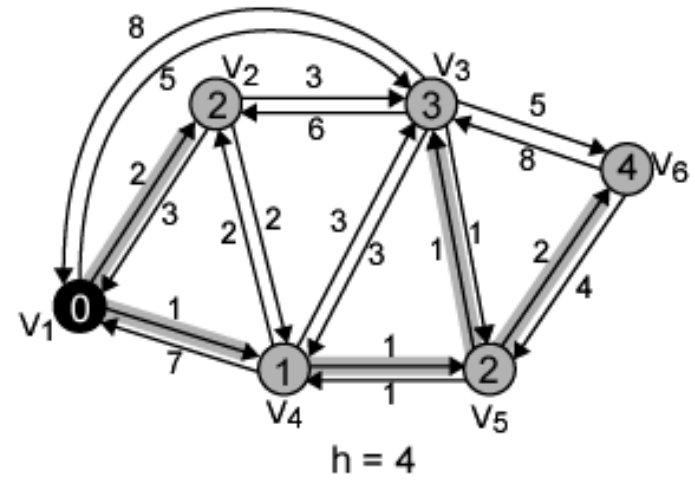
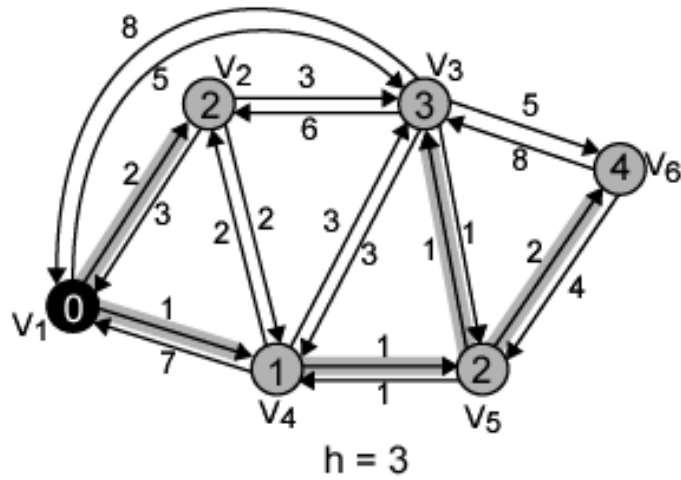
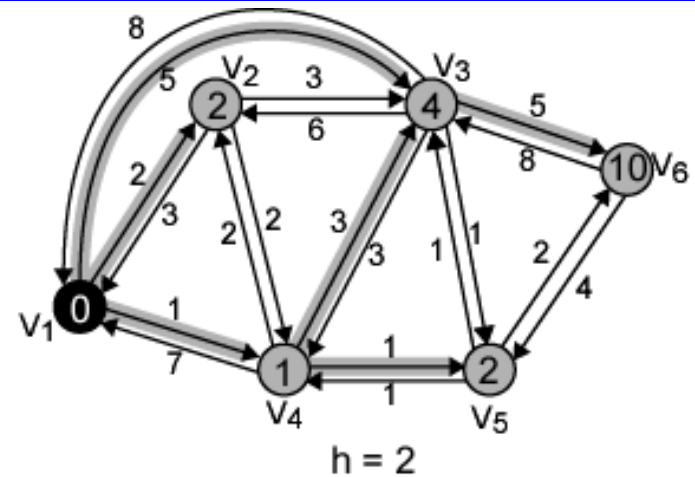
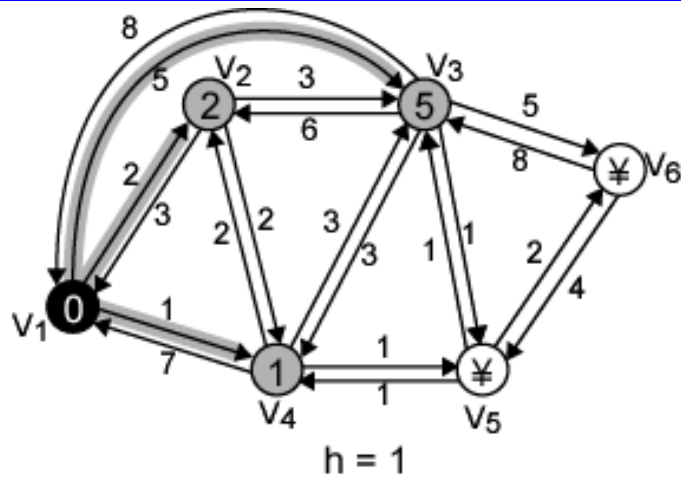
Bellman-Ford Algorithm Method

- Step 1 [Initialization]
 - $L_0(n) = \infty$, for all $n \neq s$
 - $L_h(s) = 0$, for all h
- Step 2 [Update]
- For each successive $h \geq 0$
 - For each $n \neq s$, compute
 - $L_{h+1}(n) = \min_j [L_h(j) + w(j, n)]$
- Connect n with predecessor node j that achieves minimum
- Eliminate any connection of n with different predecessor node formed during an earlier iteration
- Path from s to n terminates with link from j to n

Bellman-Ford Algorithm Notes

- For each iteration of step 2 with $h=K$ and for each destination node n , algorithm compares paths from s to n of length $K=1$ with path from previous iteration
- If previous path shorter it is retained
- Otherwise new path is defined

Example of Bellman-Ford Algorithm



Results of Bellman-Ford Example

h	$L_h(2)$	Path	$L_h(3)$	Path	$L_h(4)$	Path	$L_h(5)$	Path	$L_h(6)$	Path
0	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

Comparison

- Results from two algorithms agree
- Information gathered
 - Bellman-Ford
 - Calculation for node n involves knowledge of link cost to all neighboring nodes plus total cost to each neighbor from s
 - Each node can maintain set of costs and paths for every other node
 - Can exchange information with direct neighbors
 - Can update costs and paths based on information from neighbors and knowledge of link costs
 - Dijkstra
 - Each node needs complete topology
 - Must know link costs of all links in network
 - Must exchange information with all other nodes

Evaluation

- Dependent on processing time of algorithms
- Dependent on amount of information required from other nodes
- Implementation specific
- Both converge under static topology and costs
- Converge to same solution
- If link costs change, algorithms will attempt to catch up
- If link costs depend on traffic, which depends on routes chosen, then feedback
 - May result in instability

Required Reading

- Stalling Chapter 12
- Routing information from Comer D.
Internetworking with TCP/IP Volume 1, Prentice Hall, Upper Saddle River NJ.